Corres. and Mail
BOX AF

AF $ FRW

RESPONSE UNDER 37 C.F.R. § 1.116
EXPEDITED PROCEDURE
GROUP ART UNIT 2173

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.: 09/886,239
Filed: June 20, 2001
Inventor(s):
Jeffrey D. Washington, Michael Santori and Robert C. Young

Title: Collector Node for a Graphical Program

Examiner: Vu, Kieu D.
Group/Art Unit: 2173
Atty. Dkt. No: 5150-48900

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Commissioner for Patents, Alexandria, VA 22313-1450, on the date indicated below.

Jeffrey C. Hood

5/4/2005 (Date) JCH (Signature)

APEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed April 27, 2005, Appellants present this Appeal Brief. Appellants respectfully request that this appeal be considered by the Board of Patent Appeals and Interferences.

05/09/2005 CCHAU1 00000055 501505 09886239
01 FC:1402 500.00 DA

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expressway, Bldg. B, Austin, Texas 78759-3504, as evidenced by the assignment recorded at Reel 011955, Frame 0819.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this Appeal.

III. STATUS OF CLAIMS

Claims 1-16 are pending. Claims 1-16 are rejected and are the subject of this Appeal Brief. A copy of claims 1-16 as on appeal is included in the Claims Appendix attached hereto.

IV. STATUS OF AMENDMEMNTS

No amendments to the claims have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a method for creating a graphical program that performs a numerical function. A graphical program is a software program whose source code is not textual in nature, but rather the source code is graphical in nature, e.g., a plurality of nodes or icons connected by wires.

A node, e.g., a "collector" node in a graphical program is displayed, e.g., on a display device, such as a computer monitor, in response to user input. For example, the

node may be selected by the user from a palette provided by a graphical programming development environment, or via any of various other techniques. *See, e.g.,* Specification p. 6, lines 16-19; Figures 27 and 30-33.

The node is then configured to receive data values, also in response to user input. For example, an input terminal of the collector node may be connected to a data source in the graphical program, e.g., by connecting the input terminal to an output terminal of another node in the graphical program. *See, e.g.,* Specification p. 6, lines 20-23. The node is then configured to perform a numerical function on at least a subset of the received data values, in response to user input. For example, a graphical user interface (GUI) for configuring the collector node may be displayed in response to user input requesting to provide configuration information for the node. *See, e.g.,* Specification p. 6, lines 24-28; Figure 28. As an example, the user may specify a collection mode, i.e., under what terms the data are collected by the node, such as, for example, collecting and maintaining a moving window or "sliding block" of data, collecting a specified number of data points, i.e., a "fixed block", or collecting all data, i.e., "always". *See, e.g.,* Specification p. 6, line 28 – p. 7, line 19; Figure 28.

The graphical program is executed, during which the node receives a plurality of data values, determines a data collection on which to perform the numerical function, and performs the numerical function on the data collection, where the data collection includes at least a subset of the received data values. The node maintains state information regarding the received data values and uses the state information to determine the data collection on which to perform the numerical function. In other words, during program execution, the node collects data from the specified data source and performs the numerical function on the data, subject to the node's configuration and the received data. *See, e.g.,* Specification p. 7, lines 5-19; Figures 30-33.

In one example embodiment, the node may include one or more output terminals, each corresponding to a numerical function that can be performed on the data collection, e.g., the collector node may include an output terminal labeled as "Average" and another output terminal labeled as "Sum". In this example, the "Average" output terminal may output the average value of the data values currently being analyzed (e.g., according to the collection mode criteria specified in the GUI input panel), and the "Sum" output

terminal may output the sum of the data values currently being analyzed. *See, e.g.,* Specification p. 7, lines 20-27; Figure 30.

Independent claim 9 is directed to a memory medium that stores program instructions executable to perform the method of claim 1, the subject matter of which is summarized above.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-16 are finally rejected under 35 U.S.C. 102(e) as being anticipated by Kodosky et al. (U.S. Patent No. 5,610,828, hereinafter "Kodosky").

VII. ARGUMENT

Ground of Rejection:

Claims 1-16 are finally rejected under 35 U.S.C. 102(e) as being anticipated by Kodosky et al. (U.S. Patent No. 5,610,828, hereinafter "Kodosky"). Appellant respectfully traverses this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 2, 3, 7, 9, 10, 11, and 15

Appellant respectfully submits that each of the independent claims 1 and 9 recites one or more features not taught or suggested in Kodosky.

More specifically, Appellant submits that the cited art does not teach or suggest: "configuring the node to perform a numerical function on at least a subset of the received data values, in response to user input", nor "the node determining a data collection on which to perform the numerical function, wherein the data collection comprises at least a subset of the data values received", nor "the node performing the numerical function on the data collection", nor "wherein the node maintains state information regarding

received data values and uses the state information to determine the data collection on which to perform the numerical function".

The Examiner cites col. 15, lines 1-4, in asserting that Kodosky teaches "displaying a node in a graphical program in response to user input. Appellant notes that the cited passage describes a *structure node*, specifically, an iterative loop structure, that facilitates iterative program flow.

The Examiner then asserts that Kodosky teaches ". . . configuring the node to perform a numerical function (Minimum or Maximum) on at least a subset of the received data values, in response to user input", citing col. 18, lines 27-49, which describes Figure 27. However, the cited passage actually describes a control menu and its use in configuring a control, i.e., a graphical user interface element in a front panel, as opposed to a node in a block diagram. Appellant notes that controls do not perform "numerical functions" (e.g., an average function, a summation function, a minimum or maximum value function, etc., as described in the specification on p. 4, lines 14-15) and that the "Minimum or Maximum" in the passage cited by the Examiner refers to the specification of value bounds on data to be received or displayed by the control, *not* to max or min functions that operate to return a maximum or minimum of a pair or set of input values. Appellant respectfully submits that those of skill in the art readily understand the difference between *logical* functions and *numerical* functions, and that the Examiner has improperly attempted to equate the two.

Appellant notes that the Examiner has switched from the iterative loop structure node referred to in the Examiner's assertion regarding the first clause of claim 1, to referring to a control (as the same node) in the third clause of claim 1, although claim 1 refers only to a single node. Appellant respectfully submits that this mutually exclusive dichotomy regarding the identity of the node within the claim is improper. Applicant notes that an iterative loop structure node cannot be configured to perform a numerical function as required in the claim. Rather, an iterative loop structure node specifies iteration or looping of nodes that are contained within this node. Thus, Applicant submits that the Examiner is attempting to pick and choose different elements from the Kodosky

patent to attempt to teach the operation of "the node" in the present claims, although the claim clearly recites a single node that performs these operations. Applicant respectfully submits that this is improper. Applicant further submits that the arguments made by the Examiner are improper for the following additional reasons.

The Examiner argues that the "node" in claim 1 can be reasonably interpreted as a control, asserting that the claim does not "limit that node is in a block diagram" [sic]. Appellant respectfully disagrees, and notes that according to the Examiner's logic, the Examiner could interpret the "node" as anything not literally excluded in the claim. Appellant submits that the Examiner has improperly attempted to redefine claim terms without support, and in direct contradiction of both the cited art and the present application.

For example, Appellant notes that Kodosky, in col. 14, lines 1-2 states: "...a virtual instrument contains a front panel with a multiplicity of controls", and in col. 14, lines 11-12 states: "...a block diagram contains a multiplicity of objects of the node class", i.e., "nodes". Moreover, Kodosky clearly indicates that nodes have terminals that may couple to respective controls. For example:

> "...each object of the node class contains a multiplicity of terminals 8g. Line 8v indicates that a block diagram also contains a multiplicity of signal paths 8f. Each signal path contains a multiplicity of terminals as indicated by line 8w. There is at most one terminal per signal path that is designated as the source of the signal. Each terminal contained in a signal path also is contained in a node. However, there may be terminals in nodes which are not in signal paths. The terminals in a signal path are typically in different nodes. Lines 8y and 8z indicate that each terminal may reference a front panel control or a block diagram control (e.g., a constant). A terminal references exactly one control, and it is either on the front panel or on the block diagram. (col. 14, lines 23-36)

Nowhere does Kodosky describe or refer to a control as a node. Rather, a node may reference or couple to a control, e.g., via a terminal of the node. Furthermore, Kodosky's nodes (sometimes referred to as "icons") are *always* and *only* used in block diagrams, i.e., are *never* used in control panels. For example, with reference to Figure 88,

Kodosky states: "when a diagram is about to be executed all the nodes in the diagram are set to their armed states".

Appellant submits that there are numerous patents granted to Kodosky et al that also make this distinction, as described in the present specification. For example, regarding the content of U.S. Patent Nos. 4,901,221; 4,914,568; 5,291,587; 5,301,301; and 5,301,336; among others, to Kodosky et al, the present specification recites:

> Regarding block diagrams and nodes/icons:
> (note that the terms "icon" and "node" are used interchangeably)
>
> > "The method disclosed in Kodosky et al allows a user to construct a diagram using a block diagram editor. The block diagram may include a plurality of interconnected icons such that the diagram created graphically displays a procedure or method for accomplishing a certain result, such as manipulating one or more input variables and/or producing one or more output variables." (p. 3, lines 1-5)
> >
> > "Therefore, Kodosky et al teaches a graphical programming environment wherein a user places or manipulates icons and interconnects or "wires up" the icons in a block diagram using a block diagram editor to create a graphical 'program.'" (p. 3, lines 10-12)
> >
> > "During creation of the block diagram portion of the graphical program, the user may select various function nodes or icons that accomplish his desired result and connect the function nodes together." (p. 4, lines 3-5)
>
> Regarding front panels and controls:
>
> > "In creating a graphical program, a user may create a front panel or user interface panel. The front panel may include various user interface elements or front panel objects, such as controls and/or indicators, that represent or display the respective input and output that will be used by the graphical program". (p. 3, lines 19-22)

Regarding the distinction between controls/front panel objects and icons/nodes in the block diagram:

> When the controls and indicators are created in the front panel, corresponding icons or terminals may be automatically

> created in the block diagram by the block diagram editor. Alternatively, the user can place terminal icons in the block diagram which may cause the display of corresponding front panel objects in the front panel, either at edit time or later at run time. As another example, the front panel objects, e.g., the GUI, may be embedded in the block diagram.

Additionally, Appellant notes that the present specification nowhere describes or refers to a control as a node, and further notes that according to the specification, nodes (also referred to as "icons") are *always* and *only* used in block diagrams, i.e., are *never* used in control panels, just as in the cited Kodosky patent.

Thus, Appellant submits that nodes are specifically *not* interpretable as controls, and that such interpretation is in direct contradiction to both Kodosky and the present specification.

Appellant respectfully submits that Kodosky fails to teach or suggest "the node determining a data collection on which to perform the numerical function, wherein the data collection comprises at least a subset of the data values received", and further fails to disclose "the node performing the numerical function on the data collection".

The Examiner asserts that Kodosky teaches "executing the graphical program; the node receiving a plurality of data values during execution of the graphical program; the node determining a data collection on which to perform the numerical function, wherein the data collection comprises at least a subset of the data values received; the node performing the numerical function on the data collection; wherein the node maintains state information regarding received data values and uses the state information to determine the data collection on which to perform the numerical function", citing col. 11, lines 34-42.

Regarding the limitation "the node determining a data collection on which to perform the numerical function, wherein the data collection comprises at least a subset of the data values received", Appellant notes that Kodosky nowhere even mentions the term "collection" nor "data collection", and specifically fails to describe or disclose a node being operable to determine such a collection for the purpose of performing a numerical

function on the collection. Thus, Appellant submits that Kodosky fails to teach or suggest this limitation of claim 1.

The Examiner further cites col. 11, lines 39-42, asserting that Kodosky teaches that the conditional structure (node) performs a numerical function. However, this passage merely states that the conditional structure, being a data-flow diagram element, may use any subset of incoming signal paths, but must produce all outgoing signal paths, and thus that:

> In accordance with data-flow principles, all inputs must be available in order to start execution. Furthermore, all outputs are generated after execution is completed.

Appellant submits that the cited passage in no way supports the Examiner's assertion that the conditional structure node performs a numerical function.

In addition to the arguments presented above regarding logical and numerical functions, Appellant notes that the cited passage describes a conditional structure that may utilize any subset of incoming input signal paths, but that *does not itself perform a numerical function*. Rather, the conditional structure performs a *logical* function, and while the conditional structure may certainly include or contain nodes that perform numerical functions, the structure itself does not, and so is not properly equated with the node of claim 1.

In asserting that Kodosky teaches "wherein the node maintains state information regarding received data values and uses the state information to determine the data collection on which to perform the numerical function", the Examiner cites col. 38, lines 63-67. However, the cited passage describes *execution states* of virtual instruments (VIs), e.g., Bad, Idle, Active, Reserved, Running, Suspended, Retrying, and Error states (see col. 38, lines 21-50), as illustrated by the example execution state diagram of Figure 97. Note that these execution states indicate the general operating state of the VI, and are not related to received data values of a node, nor used to determine "the data collection on which to perform the numerical function". In other words, Appellant respectfully

submits that the Examiner has incorrectly equated Kodosky's execution states with the state information maintained by the node of claim 1.

Thus, Appellant respectfully submits that Kodosky fails to teach all of the features and limitations of claim 1, and so claim 1 and those claims dependent thereon are patentably distinct over Kodosky, and are thus allowable. Independent claim 9 includes similar limitations as claim 1, and so the above arguments apply with equal force to that claim. Thus, Appellant submits that for at least the reasons provided above, claim 9 and those claims dependent thereon are similarly patentably distinct over Kodosky, and are thus allowable.

Furthermore, Appellant respectfully submits claims 2, 3, 7, and 10, 11, and 15, dependent from independent claims 1 and 9, respectively, are also patentably distinct and allowable over the cited art, since they also include the limitations of their respective independent claims.

Claims 4 and 12

Appellant respectfully submits that in addition to the features and limitations of the independent claims, Kodosky neither teaches nor suggests "wherein the GUI enables one or more of the following collection modes to be specified for the node: Sliding Block; Fixed Block; and Always." as recited by claims 4 and 12.

For example, nowhere does Kodosky even mention a GUI operable to enable "collection modes" for a node, and specifically does not describe or disclose, or even mention, these collection modes as defined in the specification on p. 7, lines 5-19:

> If "Sliding Block" is chosen, then the data collection becomes valid once the specified number of data values has been received by the collector node during execution of the program. These data values may then be analyzed, i.e., the desired numerical function(s) may be performed on the data collection once the data collection becomes valid. From that point on, each time a new data value is received, the oldest data value is removed from the

> data collection so that the number of data values under analysis remains constant.
>
> If "Fixed Block" is chosen, then the data collection becomes valid once the specified number of data values has been received by the collector node during execution of the program. These data values may then be analyzed, i.e., the desired numerical function(s) may be performed on the data collection once the data collection becomes valid. When a new data value is then received, the old data collection is discarded, and data value collection restarts until the specified number of data values has again been received.
>
> If "Always" is chosen, then all data values are tracked during execution of the program, and the analysis is performed on all of these data values, regardless of the number of data values received.

Thus, Appellant submits that Kodosky fails to teach the features and limitations of claims 4 and 12, and thus, respectfully submits that claims 4 and 12 are patentably distinct and allowable over the cited art.

Claims 5 and 13

Appellant respectfully submits that in addition to the features and limitations of the independent claims, Kodosky neither teaches nor suggests "wherein the node is a primitive node provided by a graphical programming development environment for inclusion in the graphical program" as recited by claims 5 and 13.

For example, nowhere does Kodosky disclose a node with the functionality recited in the independent claims, and specifically does not disclose such a node being provided by graphical programming development environment as a primitive node, where "primitive node" refers to an "atomic" node intrinsic or inherent to the development environment, as opposed to a more complex node that represents a corresponding block diagram comprising further nodes, or a "custom" node designed and developed by a user using other nodes.

Thus, Appellant submits that Kodosky fails to teach the features and limitations of claims 5 and 13, and thus, respectfully submits that claims 5 and 13 are patentably distinct and allowable over the cited art.

Claims 6 and 14

Appellant respectfully submits that in addition to the features and limitations of the independent claims, Kodosky neither teaches nor suggests "wherein the numerical function performed on the data collection comprises one of: a numerical average function; a summation function; a minimum value function; a maximum value function." as recited by claims 6 and 14.

Appellant respectfully submits that Kodosky fails to disclose a node with the functionality recited in the independent claims, and specifically does not disclose such a node being operable to perform one of the numeric functions recited in claims 6 and 14, and in fact, fails to even mention an "average function", a "summation function", a "minimum function", or a "maximum function" at all. As argued above, Appellant notes that Kodosky's MINIMUM and MAXIMUM refer to setting bounds for a control, i.e., a GUI element of a front panel, *not* to the min/max numerical functions, well-known in the art of programming.

Thus, Appellant submits that Kodosky fails to teach the features and limitations of claims 6 and 14, and thus, respectfully submits that claims 6 and 14 are patentably distinct and allowable over the cited art.

Claims 8 and 16

Appellant respectfully submits that in addition to the features and limitations of the independent claims, Kodosky neither teaches nor suggests "wherein the node includes one or more output terminals corresponding to one or more numerical functions; wherein said configuring the node to perform the numerical function on at least a subset of the received data values comprises connecting a first output terminal of the node to a data target in the graphical program, wherein the first output terminal corresponds to the numerical function." as recited by claims 8 and 16.

Nowhere does Kodosky disclose a node with the functionality presented in the independent claims, and specifically does not describe such a node with output terminals, each corresponding to a respective numerical function, where connecting a particular terminal to a data target in the graphical program configures the node to perform the corresponding numerical function on data received by the node. In other words, none of the nodes described in Kodosky are operable to perform a selected one of a plurality of numerical functions on at least a subset of received data based on connection of a corresponding output terminal of the node to a data target in the graphical program.

Thus, Appellant submits that Kodosky fails to teach the features and limitations of claims 8 and 16, and thus, respectfully submits that claims 8 and 16 are patentably distinct and allowable over the cited art.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-16 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-48900/JCH.

Respectfully submitted,

Jeffrey C. Hood
Reg. No. 35,198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 5/4/2005 JCH/MSW

IX. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A computer-implemented method for creating a graphical program that performs a numerical function, the method comprising:

displaying a node in a graphical program in response to user input;

configuring the node to receive data values, in response to user input;

configuring the node to perform a numerical function on at least a subset of the received data values, in response to user input;

executing the graphical program;

the node receiving a plurality of data values during execution of the graphical program;

the node determining a data collection on which to perform the numerical function, wherein the data collection comprises at least a subset of the data values received;

the node performing the numerical function on the data collection;

wherein the node maintains state information regarding received data values and uses the state information to determine the data collection on which to perform the numerical function.

2. The method of claim 1, further comprising:

receiving user input requesting to specify configuration information for the node;

displaying a graphical user interface (GUI) for specifying configuration information for the node, in response to the user input requesting to specify configuration information for the node;

wherein said configuring the node to perform the numerical function on at least a subset of the received data values is performed in response to user input received via the GUI.

3. The method of claim 1,

wherein the GUI is useable for specifying a collection mode for the node;

wherein the collection mode determines the at least a subset of the received data values on which to perform the numerical function.

4. The method of claim 3,

wherein the GUI enables one or more of the following collection modes to be specified for the node:

Sliding Block;

Fixed Block; and

Always.

5. The method of claim 1,

wherein the node is a primitive node provided by a graphical programming development environment for inclusion in the graphical program.

6. The method of claim 1,

wherein the numerical function performed on the data collection comprises one of:

a numerical average function;

a summation function;

a minimum value function;

a maximum value function.

7. The method of claim 1,

wherein said configuring the node to receive data values comprises connecting an input terminal of the node to an output terminal of another node in the graphical program, in response to user input.

8. The method of claim 1,

wherein the node includes one or more output terminals corresponding to one or more numerical functions;

wherein said configuring the node to perform the numerical function on at least a subset of the received data values comprises connecting a first output terminal of the node to a data target in the graphical program, wherein the first output terminal corresponds to the numerical function.

9. A memory medium for creating a graphical program that performs a numerical function, the memory medium comprising program instructions executable to:

display a node in a graphical program in response to user input;

configure the node to receive data values, in response to user input;

configure the node to perform a numerical function on at least a subset of the received data values, in response to user input;

wherein during execution of the graphical program, the node is operable to:

receive a plurality of data values;

determine a data collection on which to perform the numerical function, wherein the data collection comprises at least a subset of the data values received;

perform the numerical function on the data collection;

maintain state information regarding received data values and use the state information to determine the data collection on which to perform the numerical function.

10. The memory medium of claim 9, further comprising program instructions executable to:

receive user input requesting to specify configuration information for the node;

display a graphical user interface (GUI) for specifying configuration information for the node, in response to the user input requesting to specify configuration information for the node;

wherein said configuring the node to perform the numerical function on at least a subset of the received data values is performed in response to user input received via the GUI.

11. The memory medium of claim 9,

wherein the GUI is useable for specifying a collection mode for the node;

wherein the collection mode determines the at least a subset of the received data values on which to perform the numerical function.

12. The memory medium of claim 11,

wherein the GUI enables one or more of the following collection modes to be specified for the node:

Sliding Block;

Fixed Block; and

Always.

13. The memory medium of claim 9,

wherein the node is a primitive node provided by a graphical programming development environment for inclusion in the graphical program.

14. The memory medium of claim 9,

wherein the numerical function performed on the data collection comprises one of:

a numerical average function;

a summation function;

a minimum value function;

a maximum value function.

15. The memory medium of claim 9,

wherein said configuring the node to receive data values comprises connecting an input terminal of the node to an output terminal of another node in the graphical program, in response to user input.

16. The memory medium of claim 9,

wherein the node includes one or more output terminals corresponding to one or more numerical functions;

wherein said configuring the node to perform the numerical function on at least a subset of the received data values comprises connecting a first output terminal of the node to a data target in the graphical program, wherein the first output terminal corresponds to the numerical function.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.